REMARKS/ARGUMENTS

Applicants thank the Examiner for the careful search and consideration evidenced in the subject Office Action. In light of the following remarks, Applicants respectfully request favorable reconsideration.

Claims 6 and 21 have been amended to correct a minor clerical error and place the claims in better form. In addition, the term "Native language" is changed to "Native code" to signify that a spoken language, such as Japanese or English, is not encompassed by the term. No new matter is added.

The Action rejected claims 6, 11, 12, and 14-23 as obvious in view of DeLorme et al. (U.S. Patent 6,321,158, hereinafter DeLorme) as modified using the teachings of two non-patent publications, namely, "Inside the JAVA Virtual Machine" (hereinafter "IJVM") and "Essential JNI JAVA Native Interface" (hereinafter "JNI"). Claims 6 and 21 are independent claims, and, as such, the following remarks focus primarily on those claims, since art that fails to render unpatentable a parent claim will also not render unpatentable a child claim.

In overview, while the cited art may be more or less relevant to the technological area of the invention, it fails, both alone and in combination, to teach the specifically recited elements of the claims. For example, claims 6 and 21 both recite a hybrid system wherein a first set of elements of the system is comprised of native code while a second set of elements of the system is comprised of platform independent (or multi-platform) code that interfaces with the elements of the first set. No reference shows this architecture nor indicates the desirability of such architecture.

By way of example, consider that the existence of a **first** reference describing oil and a **second** reference describing water do not amount to a **teaching** to try to mix oil and water (unless there is some hint or suggestion in the art that such a thing would have been possible and desirable). In the same way, the existence of both native code

and platform-independent code in the art does not amount to a teaching as to the desirability or possibility of a specific hybrid system combining both.

In greater detail, claim 6 recites a navigation application[1] that is implemented in native code (i.e., code native to the platform) and an optional application that is implemented in non-native code (i.e., an application that executes on a platform-independent virtual platform). An interface is recited for communicating between the navigation application and the optional application. The interface is itself also implemented in non-native code as recited.

While the Office Action has identified an "interface" in DeLorme, the noted interface of DeLorme is not an interface process *between native code and non-native code streams*. Rather, the cited interface is a data link between multiple machines. *See* DeLorme at column 8, lines 64-67. ("Alternatively, users can also operate IRMIS 100 from a remote *interface* through wireless or hard-wire links ....") *See also* column 8, lines 41-45. ("This communication *interface* between the portable PDA and home-base desktop ....") In fact, as the Action recognizes, DeLorme fails to teach non-native code (*see* page 3), so it could not teach the recited interface.

The secondary references do not solve this problem. In particular, JNI and IJVM contain many teachings related to Java, but fail to teach an interface process for communicating between running native code and running non-native code providing optional services. Thus, the cited references, singly and in combination, fail to teach the recited interface.

Of course, at first glance, with knowledge of the invention as disclosed and claimed in the patent application, it might appear that a navigation apparatus similar to the invention could be easily realized by the hypothetical modification of DeLorme in view of IJVM, as further modified by JNI. However, upon closer consideration of these references, it becomes apparent that the invention cannot be realized by the proposed modification.

---

[1] Application processing blocks are sometimes referred to here as applications to avoid excessive prose, but this terminology is simply a matter of convenient reference.

As confirmed at page 3 of JNI, the JNI generally depends upon a particular platform, i.e., platform X. "If the binary objects in a library built for platform X adhere to the JNI Application Programming Interface, then that native library can be used, without rebuilding, with any JVM that supports the JNI on platform X." Nevertheless, the JNI cannot meet the requirement of the claimed invention that the interface processing block is independent of platform. In other words, the JNI is not consistent with the interface processing block as defined in independent claims 6 and 21.

Even if a conventional interface processing block were replaced with the JNI, the optional application processing block of the claims would call the interface of the JNI directly. In this case, the JNI, which depends upon platform X, i.e., is platform dependent, must be compiled on each platform. Further, if the optional application processing block is modified during a development stage, then the JNI, i.e., the interface processing block, needs to be compiled along with the optional application processing block, based upon each platform.

By contrast with the foregoing situation, in the invention, as described in independent claims 6 and 21, the optional application processing block and the interface processing block "are executed on a virtual platform, which is executable on said platform block and another platform". With the claimed structure of the interface processing block, since the optional application processing block calls the interface processing block, the optional application processing block does not need to be compiled based upon each platform. Accordingly, when a modification to the optional application processing block occurs, the time required for compiling is decreased. Consequently, development efficiency is improved. Moreover, since the interface processing block is independent of the optional application processing block, the interface processing block does not need to be compiled upon the compiling of the optional application processing block. In other words, the interface processing block is compiled once because recompilation of the interface processing block is unnecessary, substantially improving development efficiency. This comparison shows

that neither the structure nor the advantages achieved according to the invention are suggested by the hypothetical modification upon which the rejection is based.

Moreover, the combination of references is itself not proper under the law relating to obviousness. For example, the cited modification of DeLorme in view of IJVM is said to be motivated "in order to provide users with a secure, robust, platform independent program(s) to be delivered across networks and run on a great variety of computers and devices." However, this alleged motivation does not prove the point for which it is cited. If, as the Action asserts, the desire for "robust, platform independent program(s)" would motivate one to exchange native code programs for platform independent code programs, that desire still fails to explain why one would modify DeLorme *only* in the cited respects while leaving DeLorme's navigation application implemented in native code. There is nothing in any of the references to support the asserted selective and piecemeal modification of DeLorme; in fact, the only colorable suggestion in the record for the selective reconstruction of the art appears in the applicants' own specification and claims.

Although virtually any judgment on obviousness must entail some amount of "hindsight" reasoning, such reasoning ***must only take into account*** the proven knowledge that was within the level of ordinary skill at the time the claimed invention was made; the Office ***cannot*** use knowledge gleaned from the applicant's disclosure as a basis for such hindsight reasoning. *See In re McLaughlin*, 170 USPQ 209 (CCPA 1971). *See also* MPEP 707.07.

Simply put, the references do not support the selective modification of DeLorme to first replace certain code with platform-independent code, then leave certain code as native code, and then somehow build an unspecified and untaught interface between the two, with the interface itself being platform independent. The references simply do not provide the needed elements, nor do they (or the art generally) provide the requisite motivation to complete the needed modifications.

For each of the foregoing reasons, favorable reconsideration of independent claims 6 and 21 is requested. In addition, dependent claims 11, 12, 14-20, 22, and 23

are patentable for the same reasons noted with regard to the respective independent claims.

Although the Action does not appear to invoke the doctrine of official notice expressly, if the Office feels that it has, or wishes to, affirmatively assert that the missing elements and motivation above can be supplied via this doctrine, specific documentation is respectfully requested in advance of applying that doctrine.

Prompt allowance of claims 6, 11, 12, and 14-23 is earnestly solicited.

Respectfully submitted,

Jeffrey A. Wyand, Registration No. 29,458
LEYDIG, VOIT & MAYER
700 Thirteenth Street, N.W., Suite 300
Washington, DC 20005-3960
(202) 737-6770 (telephone)
(202) 737-6776 (facsimile)

Date: March 21, 2007
JAW/PMP:ves